**Detecting Photoshopped Images with Error Level Analysis**

Luis Nova and Yufei Du

Abstract:

With technologies today, it is easier than ever to edit digital images using applications such as Adobe Photoshop or GIMP. More and more images on websites and social medias are edited, and therefore detecting edited images is becoming an important topic. Our project focuses on finding edited areas of a JPEG image using Error Level Analysis (ELA). Although currently our program is not able to perfectly detect if an image is modified or not, it is able to detect and display areas that are possibly modified.

Background:

With the advent of technologies such as digital photography and photo-editing software, the doctoring of photographs, which was once a tedious and time-consuming physical process, has become a process that anyone with a computer can engage in. As a result, fake images have become a mainstay of digital media, often depicting events that never happened or placing people (celebrities, political figures, and sports stars) in places that they never went to. These altered photos are a form of false information which, when sold as the truth, misinform the masses, which could have large-scale consequences. In order to prevent the spread of misinformation, it is important to develop technologies to counteract this manipulation through the research and development of methods in Computer Vision that can detect these attempts at doctoring photographs.

Previous Work:

There have already been a few methods developed for trying to determine if a digital photograph has been manipulated. Professor Hany Farid of Dartmouth University, a figure in the field of Digital Forensics, has contributed a few different systems in his publications, such as:

* “Exposing Photo Manipulation with Inconsistent Reflections” (J. O’Brien and H. Farid, 2012) [2.]
* “Exposing Digital Forgeries from 3-D Lighting Environments” (E. Kee and H. Farid, 2010) [3.]
* “Exposing Photo Manipulation from User-Guided 3-D Lighting Analysis” (T. Carvalho, H. Farid, and H. Farid, 2015) [4.]
* “Exposing Photo Manipulation with Inconsistent Shadows” (E. Kee, J. O’Brien, and H. Farid, 2013) [5.]

Many of these systems look for certain features in a digital image, such as shadows and reflections, and then devise models for comparing the predicted angles of reflection/casting of these features in order to see if there are any inconsistencies. Some of these systems go as far as devising full 3-Dimensional models from the extracted features for deducing where a light source might be relative to the objects in the photograph. A few of these systems require crowdsourced, user-given input on details about some of the features in the photograph that cannot be determined from pixel operations. As a result, these systems are far beyond the complexity of what a few undergrad students would be able to achieve in a few weeks’ time (though, there might be some very ambitious undergrads out there).

For the sake of implementing a system that we could test in time for the end of the semester, we chose to go with a far simpler system for detecting doctored images, based on the process of Error Level Analysis (ELA). ELA (described in detail in Neal Krawetz’s 2007 paper “A Picture’s Worth...” for Black Hat Briefings, USA) is a process that involves detecting alterations made to digital photography through the analysis of the difference in the R, G , and B color channel values between the pixels in a JPEG compressed RGB digital photo, and the pixels in the same photo passed through a certain level of more JPEG compression. The absolute value of these differences are then saved, pixel by pixel, as a new image matrix, to be saved as an image that can be opened and viewed. This new image serves as a visualization of the difference in JPEG compression levels found throughout the digital image. The difference in compression levels is a direct result of the fact that, when a layer from one image is placed onto another in software like Photoshop, that layer is usually encoded/compressed at a different level than the photo that it is being placed on. This method, while simple, is incredibly effective for producing images that highlight the difference in compression levels across a photo, which can tell an observer whether or not that photo has been manipulated [1.].

Methods:

To develop our system, we chose to write our implementation of the ELA algorithm in C++ using the OpenCV library. Our system takes in an RGB JPEG image, compresses the image using JPEG compression while retaining 95% of the original image’s quality level, and then saves the compressed image as a separate image. From here, our system iterates through each pixel in both images, taking the absolute value of the differences between the R, G, and B color channel values of both images. The new image matrix created from these differences is then observed pixel by pixel. If, at any pixel, the sum of the error level image R, G, and B channels exceeds a value of 45, then the pixel is saved as a white pixel in a new image matrix. If the sum is below 45, the pixel is saved as a black pixel in a new image matrix. This black and white binary image matrix is then blurred using OpenCV's built-in blur function, which operates on the image with a 32x32 blur-filter kernel. The final result of this process is then used for edge detection through OpenCV’s built-in canny edge detector with parameters of 1 for the first threshold for the hysteresis procedure, 3 for the second threshold, and 3 for the aperture size. After that, the result of edge detection is then passed into OpenCV’s built-in find contour function and draw contour function to draw detected regions on top of the input image matrix.

Data Set:

Our data set has 61 images. 58 of them are edited images, and 3 of them are unmodified. Edited images are from the following sources:

* The Museum of Hoaxes: A website exploring deception, mischief, and misinformation throughout history
* Gizmodo.com.au: “64 Viral Images from 2016 that were totally fake”
* Amazon.com: Product images that we could find the original background without the product
* Baidu Tieba (百度贴吧): One of the most popular discussion board in China. Users use self-edited images to make jokes.
* Screenshot of video game Kantai Collection (艦隊これくしょん): A 2D browser game that uses hand-drawn and computer-generated artworks.

We made sure that the data set includes different types of images. Here are the types of edited images included:

* Real world colored object being copied and pasted into another real world colored photo
* Real world colored object being copied and pasted into the same real world colored photo
* Real world colored photo that part of the image is re-colored
* Real world grayscale object being copied and pasted into another real world grayscale photo
* Real world colored object being copied and pasted into man-created colored image (hand drawn or computer generated)
* Real world grayscale object being copied and pasted into man-created grayscaled image
* Hand drawn colored object being copied and pasted into man-created colored image
* Hand drawn colored object being copied and pasted into real world colored photo
* Auto-generated game screenshot that has a combination of different man-created colored objects

To check if our program can detect original unmodified images, we added 3 unmodified images to the data set. They are from the following sources:

* The Museum of Hoaxes: the website has original photo beside each edited photo
* Gizmodo.com.au: similar to The Museum of Hoaxes, this website also has original photo beside each edited photo
* Photo taken by ourselves using smartphone

Results:

Out of 58 edited images, 44 of them were detected and marked with correct edited region. However, in most images, the program also marked some regions that are not actually modified. Some of these regions are noises due to over-compression of the original image, and others are regions with complicated edges and color changes. Here are some images that work well:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Input | Output | Original (If found online) |
| 1 |  |  |  |
| 2 |  |  |  |
| 3 |  |  |  |
| 4 |  |  |  |

Here are some edited images that do not work well:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Input | Output | Original (if found online) |
| 5 |  |  |  |
| 6 |  |  |  |
| 7 |  |  |  |
| 8 |  |  |  |

All 3 unmodified images were falsely detected:

|  |  |  |
| --- | --- | --- |
|  | Input | Output |
| 9 |  |  |
| 10 |  |  |
| 11 |  |  |

Overall, modifications of man-created images (1) are more accurately detected. Modifications to grayscale images (2, 4) are generally more accurate than colored images. Over-compressed images (8) are mostly inaccurate. The program also struggles to detect modifications that involve copying and pasting part of an image to itself (5, 7). It also cannot detect modifications to images that were originally taken by film camera (6, 9). Unmodified images with high contrast parts (10, 11) can also falsely trigger the detection.

Limitations:

The effectiveness of an ELA system has its limitations. For starters, ELA can only be performed on JPEG image files. This is because the features that are being checked for in ELA are a direct result of the process of JPEG compression. This means that other image formats, such as PNG images cannot be passed into our system for detection. Physically doctored photographs also will not be detected by our system, as those edits are made outside of the realm of digital values. ELA also doesn’t produce very useful results for images that have been compressed iteratively many times over, as the differences in compression levels are usually uniform throughout the image from such intense compression.

Future Work:

One of the biggest problems with building a system that takes an ELA image and tries to place bounding boxes over parts of the image that have been manipulated is that the ELA image data often is messy. While a human can distinguish parts of the image that are differently compressed, a computer will have a tough time sifting through the data without some effective thresh-holding. Due to time constraints, we were not able to effectively threshold our image beyond some basic blurring and Canny edge detection. However, we believe that we might be able to improve our system by implementing an algorithm for adaptive thresholding. We could write a method that separates the photoshopped elements in the ELA image from the non-photoshopped elements dynamically through differences in the different RGB channel values at each pixel from the average of the RGB channel values from every pixel. This would not take long to implement and test, but it would most likely be a process of trial and error.

Works Cited:

[1.] [*A Picture's Worth: Digital Image Analysis and Forensics*](http://www.hackerfactor.com/papers/bh-usa-07-krawetz-wp.pdf). Dr. Neal Krawetz, Hacker Factor Solutions, August 2007

[2.] J. O’Brien and H. Farid. Exposing Photo Manipulation with Inconsistent Reflections. *ACM Transactions on Graphics*, 31(1):4:1-4:11, 2012.

[3.] E. Kee and H. Farid. Exposing Digital Forgeries from 3-D Lighting Environments. *IEEE Workshop on Information Forensics and Security*, Seattle, WA, 2010.

[4.] T. Carvalho, H. Farid, and E. Kee. Exposing Photo Manipulation From User-Guided 3-D Lighting Analysis. *SPIE Symposium on Electronic Imaging*, San Francisco, CA, 2015.

[5.] E. Kee, J. O’Brien, and H. Farid. Exposing Photo Manipulation with Inconsistent Shadows. *ACM Transactions on Graphics*, 32(4):28:1-12, 2013.